# Du scenario linéaire aux scenarios multiples dans le projet OSSIA

Myriam Desainte-Catherine
Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France.
CNRS, LaBRI, UMR 5800, F-33400 Talence, France
myriam@labri.fr

## Contents

**Abstract**

Ce rapport présente l'étude du passage d'une représentation de scenarios linéaires à une représentation de scenarios comportant des conditions exprimant des alternatives dépendant de l'exécution en temps réel. Nous présentons d'abord le modèle temporel initial, les modèles théoriques logico-temporels proposés en début de projet, les échanges avec les utilisateurs, les problèmes à résoudre et le choix final.

# 1 Introduction

Ce rapport présente l'étude qui a conduit à l'extension du modéle de scenario linéaire du logiciel i-score vers un modèle non-linéaire. Cette étude s'est tenue durant les premiers mois du projet ANR OSSIA, confrontant scientifiques, ingénieurs et utilisateurs.

L'utilisation d'une time-line pour éditer le scenario est une contrainte du projet et aussi une orginalité permettant de mettre à la portée d'utilisateurs non programmeurs un système d'écriture puissant. La représentation interne du scenario peut donc se baser sur un arbre ou bien un graphe explicitant les relations temporelles (les arcs) entre les événements (les noeuds) composant le scenario. Le problème consistait à faire évoluer le modèle linéaire du logiciel i-score vers un modèle comportant des conditions, à la fois de façon cohérente sur le plan théorique, et satisfaisant les exigences des utilisateurs. La difficulté majeure de ce travail était dûe au fait qu'il n'existait pas d'exemples de cas pouvant être étudiés par les scientifiques. Le concept étant complètement novateur, nous ne disposions pas d'exemples de scenarios. L'étude s'est complètement construite sur les échanges entre les imaginaires des utilisateurs, des scientifiques et des ingénieurs.

La présentation de ce travail se veut intuitive et non formelle pour mettre en évidence l'esprit et la logique qui ont conduit à l'élaboration du modèle retenu.

## 1.1 Scenario arborescent linéaire

Un célèbre exemple de scenario non linéaire se trouve dans le film "Smoking / No Smoking" d'Alain Resnais qui raconte plusieurs versions d'une histoire. Le scenario est défini par un arbre binaire[1] qui représente toutes les versions de l'histoire, chaque noeud contenant une scène et une bifurcation scénaristique représentée par deux enfants et chaque arc représente la durée écoulée entre la scène parent et la scène enfant. Dans le film, ces versions sont présentées en enchaînant les scènes dans un ordre préfixe obtenu par un parcours en profondeur de l'arbre, et donc en effectuant des retours dans le passé à chaque remontée dans l'arbre. Donc au final, le scenario se présente linéairement.

## 1.2 Scenario arborescent multiple

Cependant, notre problème consiste à produire, lors de l'exécution, une seule version qui se déroule dans le temps réel conformément aux relations temporelles

[1] Voir par exemple http://nezumi.dumousseau.free.fr/film/smoking.htm

définies dans le scenario, les bifurcations dépendant de données acquises lors de l'exécution. Une version du scenario correspond donc à un chemin dans l'arbre des versions allant de la racine vers une feuille. Un modèle mieux adapté à ce problème est celui de l'*espace-temps* de la physique théorique dans lequel toues les configurations physiques naturelles sont superposées et leur évoluiton se décompose en plusieurs branches. Dans ce modèle, toutes les évolutions existent en même temps, chacune d'elles ayant sa propre amplitude d'existence et nous n'en percevons qu'une seule. L'exécution d'un tel modèle correspond plutôt à un parcours en largeur et en parallèle de l'arbre des histoires.

### 1.3 Scenario multiple non arborescent

La nature arborescente du scenario pose un problème d'explosion combinatoire dès que le nombre de cas augmente et un risque de difficulté d'édition pour l'utilisateur. Le partage de parties communes entre plusieurs versions consisterait à utiliser un graphe plutôt qu'un arbre. La concision de cette représentation éviterait la gestion de l'explosion combinatoire des cas dépliés par l'utilisateur ou le système. En conséquence, cette mise en facteur procurerait un meilleur confort d'édition.

Mais ce modèle est plus difficile à maîtriser par les utilisateurs car les différentes versions sont mélangées et partagent des événements. Cependant, la puissance d'expression est beaucoup plus interessante car elle permet de travailler soit sur un arbre comme le modèle précédent, soit de créer des parties communes et est compatible avec un modèle hiérarchique.

## 2 L'existant

À l'origine, le projet disposait du logiciel i-score basé sur un modèle de scenario linéaire à temps souple dont l'exécution était assurée au moyen d'un réseau de Petri à flux temporisé hiérarchique et coloré cadencé à la milliseconde [ADCA08, ADCLA08, ADCT11] développé lors du projet ANR Virage [AMDC$^+$10]. Le scenario était linéaire mais la souplesse du temps permettait déjà d'avoir plusieurs versions temporelles du scenario.

### 2.1 Le modèle linéaire initial

Un scenario linéaire est défini de façon hiérarchique. Il comprend les éléments suivants.

- un ensemble d'événements correspondant chacun à un début ou une fin de scenario (voir définition ci-après).
- des relations temporelles reliant deux événements, l'événement source et l'événement but. Ces relations sont des relations de précédence de points quantifiées par un intervalle de temps borné par des valeurs minimum et maximum. On appellera relations entrantes d'un événement les relations admettant cet événement pour but.
- des sous-scenarios correspondant chacun à un couple formé d'une relation temporelle et d'un processus ou d'un scenario admettant pour début l'événement source et pour fin l'événement but de la relation.

- des points d'interaction permettant d'effectuer le déclenchement lors de l'exécution en temps réel de certains événements, ces événements étant donc datés au moment de l'exécution. Ces événements sont dits interactifs.

### 2.1.1 États des événements et des relations

Une relation temporelle admet les états suivants :

- Active : son événement source a été déclenché et le temps écoulé depuis est inférieur à la borne mimimale de son intervalle de temps.
- Vérifiée : son événement source a été déclenché et le temps écoulé depuis appartient à son intervalle de temps.
- Desactivée : son événement source a été déclenché et le temps écoulé depuis est supérieur strictement à la borne maximale de son intervalle de temps.
- Inactive : en attente d'une activation.

Un événement admet les états suivants :

- Déclenchable : ses relations entrantes sont toutes vérifiées.
- Actif : l'événement est en cours de déclenchement.
- Desactivé : l'événement a été déclenché.
- Inactif : en attente d'une activation.

### 2.1.2 Ordre partiel des événements

Les relations temporelles définissent un ordre partiel entre tous les événements, qu'ils soient interactifs ou non. Lors de l'exécution, cet ordre est respecté.

## 2.2 Les propositions théoriques d'évolution

Une première étude de l'extension du modèle linéaire s'est effectuée durant la thèse de Mauricio Toro et a conduit à des propositions de modèles logico-temporels définis formellement [TDC11, TDCR14].

### 2.2.1 Des relations temporelles aux relations logico-temporelles

Une des propositions de cette thèse a consisté à étendre les relations temporelles du modèle linéaire à des relations logico-temporelles en ajoutant une condition booléenne sur chaque relation temporelle. Ainsi, pour qu'une relation logico-temporelle soit vérifiée, il faut que la condition soit vraie et que la contrainte temporelle soit vérifiée. Une relation logico-temporelle non vérifiée invalide son événement but, ainsi que la suite du scenario qui dépend de cet événement. Ce travail a conduit à l'élaboration d'une sémantique basée sur des graphes d'événements, les choix étant représentés par des conflits et utilisant la notion *d'exécution silencieuse*, prémisse de l'idée qui a conduit au modèle retenu. Par contre, aucun modèle opérationnel n'a été proposé.

### 2.2.2 Des scenarios linéaires aux scenarios conditionnels

Un modèle alternatif basé plutôt sur la notion de scenario conditionnel a aussi été proposé. Ces scenarios conditionnels comportent des conditions exclusives à l'entrée et plusieurs sous-scenarios internes qui correspondent chacun à un cas à l'instar des clauses gardées ou de la forme "cond" du langage lisp. Ils ont un point d'interaction sur leur événement de début qui reçoit lors de son activation un message permettant de déterminer le cas qui sera activé.

Les sous-scenarios sont indépendants, au sens où il n'y a pas de relations temporelles entre deux événements situés dans deux sous-scenarios différents.

L'utilisateur est assuré de construire des scenarios cohérents avec cette modélisation. Celle-ci lui permet de plus de créer des sous-versions localement tout en conservant des parties communes à toutes ses versions. La sémantique du modèle, definie hiérarchiquement en est simple ainsi que le modèle opérationnel.

# 3 De l'arbre au graphe

## 3.1 Les échanges avec les utilisateurs

Les échanges avec les utilisateurs du projet OSSIA ont bousculé les propositions théoriques et ont impliqué la recherche d'un nouveau modèle.

La proposition de scenarios conditionnels a été rejetée pour des raisons de puissance d'expression. En effet, le modèle hiérarchique interdit de poser des relations entre des sous-scenario différents, limitant ainsi l'expressivité du modèle et imposant une pratique de copie alourdissant soit l'interaction utilisateur, soit l'interface du système.

Le besoin de partage d'événements buts de plusieurs relations s'est exprimé, dans un soucis à la fois de puissance d'expression et de concision du scenario.

Ainsi, c'est un choix ambitieux et risqué qui a été fait par tous les participants du projet. Un modèle complètement général a été souhaité par les utilisateurs, avec un graphe de relations logico-temporelles pouvant relier n'importe quels événements, avec la prise de risque dûe à la difficulté, pour les utilisateurs, de maîtriser un modèle complexe et puissant, le problème de définir, pour les scientifiques, un modèle opérationnel simple pour l'exécution d'un scenario représenté par un graphe et la contrainte, pour les ingénieurs, de pouvoir faire évoluer le code existant de façon simple et fiable.

## 3.2 Cas critiques de scenarios non arborescents

Dans le cadre d'un scenario conditionnel, les événements ayant plusieurs relations entrantes posent un sérieux problème. Doivent-ils attendre toutes les relations temporelles, ou seulement celles qui correspondent à la version qui est active? Comment éviter les blocages dus à l'attente d'une relation non active?

Pour illustrer ces problèmes, prenons l'exemple suivant qui nous servira tout au long de ce rapport. Soient deux conditions booléennes `c1` et `c2` indépendantes portées par des relations ayant pour buts respectivement des événements `e1` et `e2`. Si l'utilisateur construit un événement `e` admettant `r1` comme relation entrante provenant de `e1`, le déclenchement de `e` est conditionné par la verification de la relation `r1` et par le déclenchement de `e1`, conformément au modèle

initial. Mais si l'utilisateur construit en plus une relation temporelle `r2` allant de `e2` vers e, les questions suivantes se posent :

- l'événement `e` doit-il attendre que `r1` et `r2` soient vérifiées, auquel cas il s'agit d'un mécanisme de conjonction.
- l'événement `e` doit-il attendre la vérification de l'une des deux relations seulement, auquel cas il s'agit d'un mécanisme de disjonction.
- si plus de deux conditions sont considérées, les cas sont plus nombreux et difficiles à exprimer.

### 3.2.1 Risques de blocage

Dans le cas de la conjonction, le déclenchement de l'événement `e` se produit quand `r1` et `r2` sont vérifiées. Cependant, étant donné que le résultat des conditions `c1` et `c2` est imprédictible statiquement, tous les cas suivants peuvent se produire :

- `c1` est vraie et `c2` est fausse : l'événement `e2` ne sera pas déclenché, et en conséquence la relation `r2` ne sera jamais vérifiée. Si l'événement `e` attend la vérification de la relation `r2`, il ne s'activera jamais, bloquant ainsi toute la suite du scenario.
- `c1` est fausse et `c2` est vraie : même risque de blocage.
- `c1` est fausse et `c2` est fausse : un blocage se produira si l'événement `e` attend la vérifiaction des 2 relations `r1` et `r2`.
- `c1` est vraie et `c2` est vraie : c'est le seul cas qui ne provoque pas de blocage.

En conséquence, ce mécanisme permet d'activer l'événement `e` si et seulement si toutes les relations sont actives. Dans tous les autres cas, un blocage se produit empêchant l'exécution de la suite du scenario. Ainsi, toutes les versions du scenario partageant une partie de cette suite se trouveront bloquées même si elles sont actives.

### 3.2.2 Risques d'exécutions multiples

Dans le cas de la disjonction, on souhaite qu'une des relations puisse déclencher e. Nous avons les cas suivants :

- Si l'une des conditions `c1` ou `c2` est vraie, un seul des événements `e1` ou `e2` et une seule des relations `r1` ou `r2` sont activés. L'événement sera activé lors de la vérification de la relation activée, et tout se passera conformément à ce qui est attendu.
- Si les deux conditions sont vraies, les deux événements `e1` et `e2` sont activés ainsi que les relations `r1` et `r2`. Le résultat risque de dépendre de l'ordre dans lequel les relations `r1` et `r2` sont vérifiées. Si la première permet le déclenchement de e, la deuxième rique de déclencher `e` une deuxième fois en décalé, ou bien produire des erreurs si les processus ne supportent pas une deuxième exécution.

- Si aucune des deux conditions n'est vraie, aucune des relations ne sera activée et l'événement `e` ne sera donc pas activé, ce qui est correct.

Le mécanisme de disjonction présente donc des risques d'exécutions mutiples des parties de scenarios qui seraient partagées par plusieurs versions du scenario. En effet, des versions indépendantes actives peuvent déclencher plusieurs fois en séquence ces sous-parties comme dans l'exemple ci-dessus, puisqu'aucune synchronisation n'est effectuée sur l'événement e.

#### 3.2.3 Choix utilisateur / choix système

On voit que le choix entre conjonction et disjonction comporte des risques importants et qu'il nécessite une expertise de l'utilisateur. Or, dans notre projet, l'utilisation du système doit être simple et intuitive et il est impératif de ne pas mettre l'utilisateur devant une telle responsabilité. La voie que nous avons adoptée consiste plutôt à faire un choix par défaut au niveau du système, quitte à réduire dans un premier temps l'expressivité du modèle pour que celui-ci évite les risques évoqués et puisse être facilement maîtrisé par l'utilisateur. Son implementation a consisté à exploiter la couleur des jetons du réseau de Petri.

# 4 Le modèle retenu

Basé sur la théorie des mondes multiples, le modèle retenu permet de relier de façon cohérente l'écriture et l'exécution du scenario car le modèle de temps décrit la dynamique de façon statique. Il a nécessité la construction d'un nouvel objet qui est l'événement conditionnel.

## 4.1 La théorie des mondes multiples

Cette théorie issue de la physique théorique introduit l'indéterminisme dans l'espace-temps. Selon cette théorie, plusieurs mondes sont superposés, au sens quantique du terme. Chaque monde est composé de configurations physiques naturelles et de leur évolution. Chacune de ces évolutions, appelée *histoire* possède une amplitude d'existence, qui peut se concevoir comme une couleur. Deux particules sont de la même couleur, c'est-à-dire dans le même monde, si elles sont en interaction. Pour passer d'une configuration à une autre, toutes les couleurs de toutes les histoires sont mélangées. Ainsi, nous avons l'illusion d'une seule réalité, alors que plusieurs autres réalités se déroulent hors de notre propre monde [DC].

Cette théorie constitue une excellente source d'inspiration pour faire évoluer notre modèle vers l'indéterminisme. En effet, elle est complètement compatible avec l'indéterminisme temporel déjà présent dans le modèle initial [DCAA13] : tous les cas sont présents dans la partition, mais une seule version s'exécute.

## 4.2 Les événements interactifs conditionnels

Un événement interactif conditionnel est un nouvel objet dans le modèle qui va servir à construire les chemins des différentes versions. L'événement interactif conditionnel est isolé et ne correspond ni à un début ni à une fin de sous-scenario. Un événement conditionnel comporte plusieurs clauses gardées, qui

sont évaluées en séquence dans le même tic d'horloge, lors du déclenchement de l'événement. Ces clauses comportent chacune une condition booléenne en prémisse et une ou plusieurs relations en conséquence. Ainsi, l'exécution d'une clause consiste à évaluer la condition puis à activer la ou les relations si celle-ci est vraie.

Les utilisateurs ont souhaité que les clauses ne soient pas exclusives. Donc contrairement à la forme "cond" du langage lisp, toutes les clauses sont évaluées quelque soient les résultats des conditions. Ainsi, n'importe quelle configuration de déclenchements est permise.

## 4.3 Le modèle opérationnel

Ce modèle consiste à considérer que toutes les versions d'un scenario s'éxécutent en même temps avec des amplitudes d'existence propres. Comme dans la théorie physique des mondes multiples, tous les cas existent mais seuls ceux qui sont en interaction sont dans le même monde. Dans notre modèle, cela signifie que les événements reliés par une relation temporelle sont dans une même version. De plus, toutes les versions s'exécutent, mais une seule est perçue lors d'une exécution donnée. Les événements de la version perçue sont attribués d'une amplitude d'existence de 1 et les autres d'une amplitude de 0. Une exécution consiste donc à parcourir le graphe du scenario en propageant en temps réel les amplitudes d'existence en fonction des relations temporelles qui sont valides.

L'implémentation du modèle initial est basée sur un réseau de Petri temporisé coloré et hiérarchique. Le réseau étant donné en statique, seuls les jetons portent des informations dynamiques. L'idée de ce modèle est donc d'utiliser les couleurs des jetons pour porter les amplitudes d'existence.

On obtient deux types de jetons :

- les jetons actifs qui activent pleinement les événements et les relations tempolrelles,
- les jetons passifs qui ne font que passer sans déclencher les événements ni exécuter les processus et ni attendre les intervalles de temps des relations temporelles.

Ainsi toutes les realités s'exécutent, mais seule la réalité perçue effectue pleinement les calculs.

## 4.4 Un nouvel état

Il est nécessaire de considérer un nouvel état correspondant aux événements et relations traversés par des jetons passifs. Ainsi, on définit l'état *invalide* de la façon suivante :

- Une relation se trouvant dans une clause dont la condition est fausse est invalide.
- Un événement n'ayant que des relations entrantes invalides est invalide.
- Une relation ayant pour source un événement invalide est invalide.
- Un événement ayant au moins une relation entrante vérifiée est declenché.

## 4.5 Une conjonction pour simuler une disjonction

Avec un tel modèle d'exécution, les événements admettant plusieurs relations entrantes sont déclenchés lorsque toutes les relations actives sont vérifiées. Ce fonctionnement simule dans la réalité perçue une disjonction puisque l'événement but sera activé dans tous les cas sauf si toutes ses relations entrantes sont invalidées.

Ainsi, la conjonction effective lors de l'exécution apporte la synchronisation qui permet d'éviter le risque d'exécutions multiples. Et la disjonction perçue dans la réalité permet de partager des parties du scenario sans se soucier de blocages éventuels dûs aux synchronisations avec des versions non actives.

## 4.6 Cas critiques

Reprenons l'exemple précédent pour illustrer le comportement du système sur les cas critiques. Tout d'abord il faut modéliser ce cas au moyen de nos événements conditionnels.

Supposons que les conditions `c1` et `c2` soient portées par des clauses d'un événement conditionnel. On considère deux relations temporelles conséquences de ces deux clauses, l'une associée à `c1` et allant vers `e1` et l'autre associée à `c2` et allant vers `e2`. On a de plus une relation `r1` allant de `e1` vers `e` et une relation `r2` allant de `e2` vers e.

Lors de l'exécution les cas suivants peuvent se produire :

1. Les deux conditions `c1` et `c2` sont vraies : les deux événements `e1` et `e2` sont activés lors de la vérification de leurs relations temporelles entrantes. Puis `e` est déclenché lors de la vérification des relations `r1` et `r2`, comme dans le modèle linéaire on a une véritable conjonction synchronisée.

2. `c1` est vraie et `c2` est fausse : l'événement `e1` est activé lors de la vérification de sa relation entrante, mais `e2` est invalidé. La relation `r2` s'en trouve invalidée et `e` est déclenché lorsque la relation `r1` est vérifiée. La condition `c1` suffit donc à déclencher `e` comme dans une disjonction.

3. `c1` est fausse et `c2` est vraie : le cas symétrique marche tout aussi bien.

4. `c1` est fausse et `c2` est fausse : les relations `r1` et `r2` sont invalidées, ce qui ne déclenchera pas e. L'évenement `e` sera invalide s'il n'a pas d'autres relations entrantes qui ne soient pas invalides.

### 4.6.1 Risques de blocages

Ce système permet d'éviter les blocages dûs à des conditions fausses. En effet, les événements non activés sont traversés par des jetons passifs, ce qui évite d'attendre les relations non activées. De plus, le système permet une édition fiable et incrémentale. Ajouter une relation temporelle vers un événement but n'apporte aucun risque de blocage quelle que soit la configuration du graphe.

### 4.6.2 Risques d'exécutions multiples

Le problème des exécutions multiples se rapporte à la disjonction de relations temporelles ayant le même but et dont la vérification ne se produit pas en même

temps. L'exécution multiple se produiit si au moins deux relations sont vérifiées, ce qui implique deux activations en décalé de l'événement but.

Dans notre modèle, le mécanisme de conjonction mis en place implique l'attente de la vérification de toutes les relations actives, ce qui nous prémunit d'un déclenchement précoce de l'événement. Les relations invalides étant traversées par des jetons qui ne prennent pas de temps ne posent pas de problème d'attente.

### 4.6.3 Choix utilisateur / choix système

Ce mécanisme ne nécessite aucun choix de la part de l'utilisateur. Le choix essentiel fait par le système est d'utiliser un mécanisme par défaut de conjonction simulant la disjonction qui évite complètement les problèmes de blocages et d'exécutions multiples. Cependant, l'expressivité du langage s'en trouve diminuée au profit de la fiabilité. Une véritable conjonction n'est pas exprimable dans ce modèle. Mais comme indiqué en perspective, des solutions simples existent pour la mettre en place.

# 5 Perspectives

Concernant l'expressivité du langage, il serait intéressant dans des versions ultérieures de mieux exploiter les amplitudes d'existence et de généraliser leur mélange. Dans le modèle actuel, un jeton actif suffit à déclencher un événement. On pourrait paramétrer ce comportement pour utiliser par exemple la somme des amplitudes d'existence ce qui permettrait à l'utilisateur de définir combien de jetons doivent être reçus par un événement pour être déclenché. On pourrait aussi exploiter la couleur des jetons et utiliser des amplitudes d'existence comprises entre 0 et 1 ou des nombres complexes et de paramétrer aussi le seuil de perception des versions du scenario.

# 6 Conclusion

Nous avons présenté une extension du modèle de scenario linéaire du logiciel i-score vers un modèle non linéaire en introduisant de la logique. Nous avons développé l'étude scientifique des propositions des utilisateurs et de leurs risques et nous avons proposé un nouveau modèle permettant d'étendre simplement le modele linéaire initial.

Ce modèle est une des innovations majeures du projet OSSIA. Il a permis de répondre pleinement aux exigences des utilisateurs, tout en permettant une évolution cohérente et simple du modèle. Il allie de plus puissance d'expression, fiabilité et simplicitié d'utilisation.

# References

[ADCA08] Antoine Allombert, Myriam Desainte-Catherine, and Gérard Assayag. Iscore : Writing the interaction. In *Proceedings of the 3rd Digital Interactive Media in Entertainment and Art (DIMEA), Athens, Greece*. September 2008.

[ADCLA08] A. Allombert, M. Desainte-Catherine, J. Larralde, and G. Assayag. A system of interactive scores based on qualitative and quantitative temporal constraints. In *Proceedings of ARTECH 2008 the Fourth International Conference on Digital Arts,Porto, Portugal*. 2008.

[ADCT11] Antoine Allombert, Myriam Desainte-Catherine, and Mauricio Toro. Modeling temporal constraints for a system of interactive score. In *Constraint Programming in Music*, pages 1–23. Wiley, 2011.

[AMDC+10] Antoine Allombert, Raphaël Marczak, Myriam Desainte-Catherine, Pascal Baltazar, and Laurent Garnier. Virage : Designing an interactive intermedia sequencer from users requirements and the background. In *International Computer Music Conference, New York, USA*. june 2010.

[DC] Thibault Damour and Jean-Claude Carrière. *Entretiens sur la multitude du monde*. Odile Jacob.

[DCAA13] Myriam Desainte-Catherine, Antoine Allombert, and Gérard Assayag. Towards a hybrid temporal paradigm for musical composition and performance: The case of musical interpretation. *Computer Music Journal*, 37(2):61–72, 2013.

[TDC11] Mauricio Toro and Myriam Desainte-Catherine. Concurrent constraint conditional branching interactive scores. In *Sound and Music Computing (SMC)'10. Barcelona, Spain*, pages 270–273. july 2011.

[TDCR14] Mauricio Toro, Myriam Desainte-Catherine, and Camilo Rueda. Formal semantics for interactive music scores: a framework to design, specify properties and execute interactive scenarios. *Journal of Mathematics and Music*, Volume 8, Issue 1:93–112, 2014.